Day 3: Toboggan Trajectory

(Povezava na nalogo]

V tretji nalogi je podan zemljevid gozda.

```
..##.......
#...#...#..
.#....#..#.
..#.#...#.#
.#...##..#.
..#.##.....
.#.#.#....#
.#........#
#.##...#...
#...##....#
.#..#...#.#
```

To je zgolj začetek - isti vzorec se še velikokrat ponovi proti desni. Z zgornjega levega oglišča se spustimo desno dol, tako da gremo v vsakem koraku vrstico nižje in tri znake desno.

```
O.##.........##.........##.........##.........##.........##.......  --->
#..O#...#..#...#...#..#...#...#..#...#...#..#...#...#..#...#...#..
.#....X..#..#....#..#..#....#..#..#....#..#..#....#..#..#....#..#.
..#.#...#O#..#.#...#.#..#.#...#.#..#.#...#.#..#.#...#.#..#.#...#.#
.#...##..#..X...##..#..#...##..#..#...##..#..#...##..#..#...##..#.
..#.##.......#.X#.......#.##.......#.##.......#.##.......#.##.....  --->
.#.#.#....#.#.#.#.O..#.#.#.#....#.#.#.#....#.#.#.#....#.#.#.#....#
.#........#.#........X.#........#.#........#.#........#.#........#
#.##...#...#.##...#...#.X#...#...#.##...#...#.##...#...#.##...#...
#...##....##...##....##...#X....##...##....##...##....##...##....#
.#..#...#.#.#..#...#.#.#..#...X.#.#..#...#.#.#..#...#.#.#..#...#.#  --->
```

Znaki `O` in `X` predstavljajo našo pot - `O` pomeni proste celice, `X` pa tiste, kjer zadanemo drevo. Zanima nas, koliko dreves bomo podrli.

Prvi del

Branje je trivialno.

```
forest = [x.strip() for x in open("input.txt")]
```

Zdaj pa spust. Naredili bomo zanko čez vrstice, `for line in forest`. Poleg tega bomo hranili tudi koordinato `y`, saj je iz nje trivialno izračunati `x`: `x = y * 3`. Koordinato `y` bomo dobili kar z `enumerate`.

```
hits = 0
for y, line in enumerate(forest):
    if line[y * 3] == "#":
```

```
        hits += 1
print(hits)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-27-eabe8b28dd4a> in <module>
      1 hits = 0
      2 for y, line in enumerate(forest):
----> 3     if line[y * 3] == "#":
      4         hits += 1
      5 print(hits)

IndexError: string index out of range
```

Seveda ne. `y * 3` je hitro prevelik, gozd pa se ponavlja. To, da "*pridemo okoli*" najlažje opišemo z ostankom pri deljenju z dolžino vrstice.

```
n = 0
for y, line in enumerate(forest):
    if line[y * 3 % len(line)] == "#":
        n += 1
print(n)
```

```
7
```

Po zgledu tega, kar smo se naučili ob prejšnji nalogi, spremenimo to v vsoto: šteti želimo, kolikokrat je bil izraz `line[y * 3 % len(line)] == "#"` resničen, torej enak `True` (`1`) in ne `False` (`0`).

Drugi del

Drugi del od nas zahteva, da preverimo različne smeri, konkretno

```
directions = [(1, 1), (3, 1), (5, 1), (7, 1), (1, 2)]
```

pri čemer prvo število pomeni dolžino pomika navzdol, drugi dolžino pomika desno.

Gornji program spremenimo v funkcijo in mu dodamo argumenta `dx` in `dy`.

```
def hits(dx, dy, forest):
    n = 0
    for y, line in enumerate(forest[::dy]):
        if line[y * dx % len(line)] == "#":
            n += 1
    return n

hits(3, 1, forest)
```

```
7
```

Kam sta šla `dx` in `dy`? Za `dx` je očitno - nadomestil je trojko. `dy` pa je razredčil gozd: s `forest[::dy]` poskrbimo, da bomo gledali samo vsako `dy`-to vrstico gozda.

Naloga pravi, da je potrebno izračunati število zadetih dreves za vsako od podanih smeri in to namnožiti skupaj.

```
product = 1
for dx, dy in directions:
    product *= hits(dx, dy, forest)
print(product)
```

```
2431272960
```

Krajša rešitev

Tako tole rešimo z generatorji.

Očitno bi bilo to mogoče natlačiti v eno samo vrstico. A to ni poanta. Tole je še vedno berljivo.

```
from functools import reduce
from operator import mul

forest = [x.strip() for x in open("input.txt")]
directions = [(1, 1), (3, 1), (5, 1), (7, 1), (1, 2)]

def hits(dx, dy, forest):
    return sum(line[y * dx % len(line)] == "#"
               for y, line in enumerate(forest[::dy]))

print(hits(3, 1, forest))

print(reduce(mul, (hits(dx, dy, forest) for dx, dy in directions)))
```

```
184
2431272960
```